Lab 1 Solution:

1.

public class ASingleton{  
 private static ASingleton *uniqueClass* = null**;** private static Object *lock* = new Object()**;** public static ASingleton getInstance(){  
 if(*uniqueClass* == null){  
 synchronized (*lock*){  
 if(*uniqueClass* == null){  
 *uniqueClass* = new ASingleton()**;** }  
 }  
 }  
 return *uniqueClass***;** }  
}

I use **double check lock** to create singleton with thread safe. I check null for uniqueClass variable two times to make sure it only instantiate uniqueClass when it is null and synchronized keyword to make source code in that block runs only in one thread at the same time. It creates thread safe.

2.

There is a simpler way to implement singleton with enum. Java enum value is instantiated only once in Java Program.

public enum enumSingleton{  
 *INSTANCE***;** enumSingleton(){}  
 public void doSomething(){  
 }  
}